LECTURE 16

WHAT FUNCTIONS $K(x,y)$ ARE KERNELS?

- Can we express $\forall$ function $K(\cdot,\cdot)$ as $K(x,y) = \langle \phi(x), \phi(y) \rangle$?
- No: the matrix $\left[K(x_i, x_j) \right]_{i,j=1}^{n} = \left[\langle \phi(x_i), \phi(x_j) \rangle \right]_{i,j=1}^{n}$ is a Gram matrix $\Rightarrow$ must be symmetric PSD: $K \succeq 0$. (HW8)
- This necessary condition is also ≈sufficient:

THM (Mercer) Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a symmetric, continuous function such that

$$\left[K(x_i, x_j) \right]_{i,j=1}^{n} \succeq 0 \quad \forall\, x_1, \dots, x_n \in \mathbb{R}^d.$$

Then $\exists$ Hilbert space $\mathcal{H}$ and a map $\phi : \mathbb{R}^d \to \mathcal{H}$ such that

$$K(x,y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \quad \forall\, x, y \in \mathcal{H}.$$

"feature space" — $\mathcal{H}$

"feature map" — ϕ

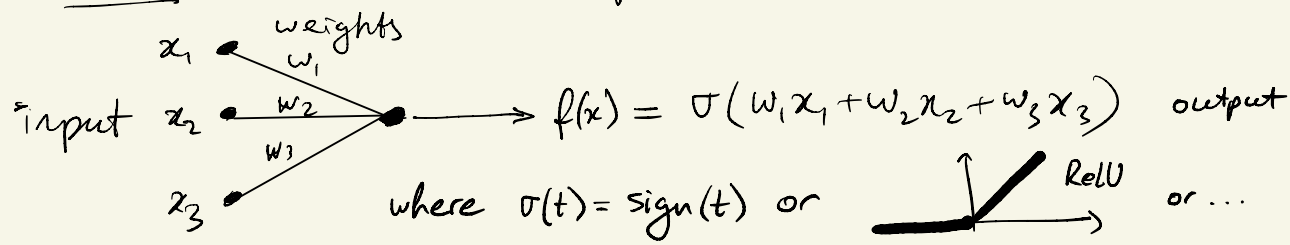
[This is an ∞-dimensional version of the HW8 problem:
$\forall$ PSD matrix is a Gram matrix $(x \mapsto i,\ y \mapsto j)$]

- Mercer's condition (PSD) is difficult to check. However, using it, we can build new kernels from old ones using simple rules (HW), such as addition, exponentiation, etc.
- Examples: (a) RBF $K(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{\ell} \right)$, (b) polynomial $K(x,y) = (1 + \langle x, y \rangle)^p$, ... HW

Kernels arise in:

NEURAL NETWORKS

Def A *neuron* is a composition of linear & nonlinear function

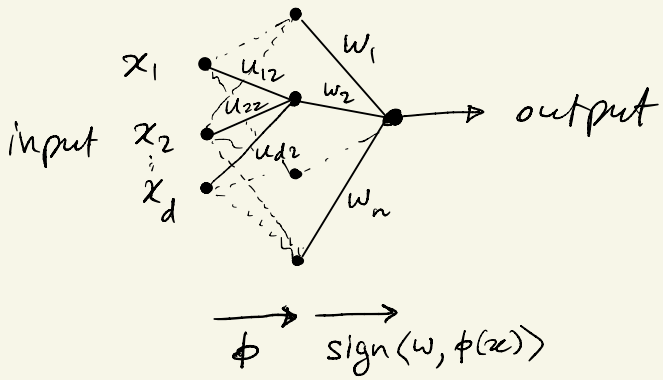


Ex: SVM is a neuron: $\text{sign}\langle w, x\rangle = \text{sign}(w_1x_1 + \dots + w_dx_d)$

We train weights w_j

Def A *neural network* is a superposition of neurons.

We train weights (u_{ij}), (w_j).



Connection to kernels:

- The first layer computes a *feature map* $\phi: \mathbb{R}^d \to \mathbb{R}^n$:

$$\phi(x) = \begin{bmatrix} \text{sgn}\langle u_1, x\rangle \\ \dots \\ \text{sgn}\langle u_n, x\rangle \end{bmatrix}$$

- *Kernel*:

$$K(x,y) = \langle \phi(x), \phi(y)\rangle = \frac{1}{n}\sum_{i=1}^{n} \text{sgn}\langle u, x\rangle\langle u, y\rangle \xrightarrow{n\to\infty}$$

Let's initialize all weights $u_{ij} \sim N(0,1)$ iid

$$\xrightarrow{n\to\infty} \mathbb{E}\,\text{sgn}\langle u, x\rangle \text{sgn}\langle u, y\rangle \quad \text{where } u \sim N(0, I_n)$$

$$= \frac{2}{\pi}\arcsin\langle x, y\rangle.$$

Hence: a 1-layer neural network $\simeq$ kernel SVM.

- **But**: after initialization, a neural network is *training* weights u_{ij}

$\Rightarrow$ trains kernel K.

- Dynamics of the prediction function is described by the *neural tangent kernel* (NTK).

A REFRESHER IN LINEAR ALGEBRA

SPECTRAL DECOMPOSITION

SEE MORE IN HDP-2022

Thm (Spectral thm for symmetric matrices)

$\forall$ symmetric $n \times n$ matrix A,

(a) all eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ are real;

(b) the unit eigenvectors $u_1, \ldots, u_n$ are orthonormal ($\Rightarrow$ o.n. basis)

Cor (Spectral decomposition) $\forall$ symmetric $n \times n$ matrix A:

$$A = \sum_{i=1}^{r} \lambda_i u_i u_i^T \quad \text{where } r = \text{rank}(A)$$

and (λ_i, u_i) are eigenvalues & unit eigenvectors of A.

SKIP

$\forall x \in \mathbb{R}^n$: $x = \sum_{i=1}^{n} \langle u_i, x \rangle u_i$ (o.n. basis decomposition)

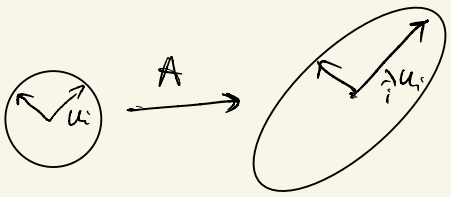
$\Rightarrow Ax = \sum_{i=1}^{n} \langle u_i, x \rangle \underbrace{Au_i}_{\lambda_i u_i} = \underbrace{\left(\sum \lambda_i u_i u_i^T\right)}_{B} x$

$\Rightarrow Ax = Bx \ \forall x \overset{\text{why?}}{\Longrightarrow} A = B.$

(a) Matrix form:

$$A = U \Lambda U^T$$

(b) Geometric form



(c) Optimization form:

$\lambda_1 = u_1^T A u_1 = \max_{x \text{ unit}} x^T A x$, $u_1 = \text{argmax} \ldots$ (λ_1 times)

$\lambda_2 = u_2^T A u_2 = \max_{x \perp u_1, \text{ unit}} x^T A x$, $u_2 = \text{argmax} \ldots$

$\lambda_3 = u_3^T A u_3 = \max_{x \perp (u_1, u_2), \text{ unit}} x^T A x$, $u_3 = \text{argmax} \ldots$

$\ldots$

$\lambda_n = u_n^T A u_n = \min_{x \text{ unit}} x^T A x$, $u_n = \text{argmin} \ldots$